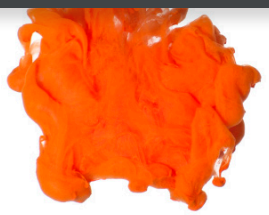
cih Solutions
SERVICE MANAGEMENT INTEGRATORS

THE ENDURING MYTH OF CMDB



This white paper discusses how a CMDB can be used as the basis of a Configuration Management System to manage risk and control costs in an IT Service Management environment. The paper identifies four 'hot spots' based on the author's experience and outlines common problems and suggests solutions using a CMDB.experience, while keeping the IT service costs under control.

Introduction

Although entitled 'The Enduring Myth of CMDB' this white paper is actually about configuration management. My aim is to show that although a CMDB (Configuration Management Data Base) is an essential component of a robust Configuration Management System (CMS), there are other key components that drive best practice, namely people and processes. The paper will look at a definition of CMDB, how to select a CMDB solution, how to populate it and especially how to make it work within a Configuration Management System. It will also look at the role of a CMDB in two key ITIL processes – Change and Incident Management.

But what exactly do I mean by 'myth' in this instance - well, I'll come to that shortly but first we need to consider what is actually meant by a CMDB. Let's be clear at the outset - configuration management is nothing new.

Whatever construct of IT system you have, the IT operational staff will need to record all the building blocks - known as Configuration Items (CI) - and record where the CIs are located, how they are configured and - critically - how they work together.

Basically, a CMDB provides a central repository for all CIs in the form of CI Records and makes it available to the IT organisation and to the ITSM team to support change management and other ITSM processes.

"When I first encountered the ITIL i-CMDB concept my immediate reaction was - this can't be done".

So, is a CMDB needed? Actually yes it is, but what type of CMDB? In fact, is there more than one type? As our starting point I want to look at what ITIL says on the subject, and work back from there. A quick look at the ITIL v3.0 schematic diagram for the Service Asset and Configuration Management (SCAM) function shows an Integrated CMDB at the core of the SACM. For convenience, I will just call this i-CMDB (integrated CMDB).

To be honest, when I first encountered the ITIL i-CMDB concept my immediate reaction was - this can't be done.

As yet, I'm not aware of any IT organisation that has fully implemented the SACM as defined by ITIL, but there are some very useful guidelines on what is actually required to implement an i-CMDB in a Gartner report titled Critical Capabilities for Configuration Management Database, published in 2014[1].

ITIL and the i-CMDB

Basically, the design of an i-CMDB is more than just a repository for configuration data. It is in fact a metadatabase that does not store the CI data but connects to several different data sources associated with the CIs. This is known as federation. The i-CMDB also manages issues around CI reconciliation and synchronisation; and also models and maps logical CI configurations as a visible graphic for the end users.

It's worth taking a closer look at four of the critical capabilities to see what is involved. The graphic shown below in Figure 1 is a simple representation of how the i-CMDB is at the core of the ITIL SACM and 'integrates' all the IT Asset Management (ITAM) CI landscape with the ITSM processes to create an end-to-end Configuration Management System.

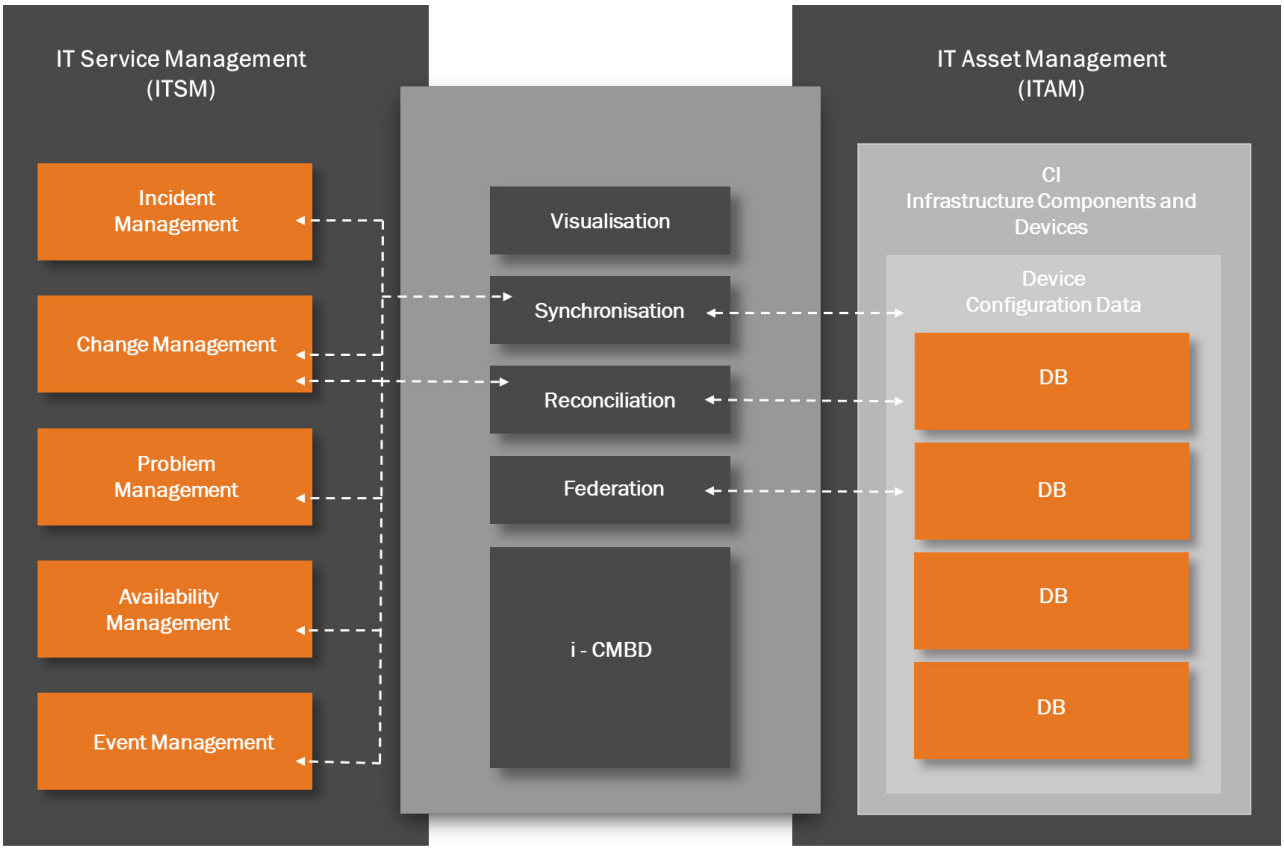


Figure 1 - An Integrated CMDB based on ITIL v3.0

The four capabilities shown in **Figure 1** are:

Federation - this is the capability to pull configuration data from several different CI database sources. This is not as easy as it might appear. The big issue here is that a mechanism is needed to ensure that two CI configuration data sources reference the same data. Which of the data sources is the one to use? Which one is the authoritative source? This is why reconciliation is needed.

Reconciliation - this is the capability that ensures there is no duplicate CI data. This is achieved by reconciling the location and attributes of the CIs. Reconciliation must also adjust data extracted from more than one CI data source to eliminate duplication and maintain consistency of CI data.

However, this presents something of a conflict. If the reconciliation engine creates a change in one of the CI data sources then this change needs to be identified as an approved or a non-approved change. This is done by synchronisation between the federated CI databases.

Synchronisation - this is the capability to maintain synchronisation with all the CI databases. The i--CMDB will need to identify changes made to the infrastructure and then distinguish between approved changes and non-approved changes. Approved changes will be made via the Change Management process for which a Request for Change (RFC) has already been raised. The synchronisation engine will compare the detected change against an approved list of changes and then raise an alert if an unauthorised change is detected.

Visualisation - this is the capability to model and map all the CI configurations that form the IT infrastructure landscape and present these as a set of logical diagrams that show CI relationships and interconnections in a graphical form that aids ITSM change management personnel with assessing the impact of a proposed change.

So, why do I think the ITIL approach to a CMDB won't work? There are a number of reasons that come to mind. Is it really possible to automate Reconciliation and Synchronisation and be certain that what you finally see is in fact an accurate snap shot of the current system configuration? One example is the recommended use of auto-discovery tools for regular daily updates to configuration. Whilst these tools have some useful features what happens when an unauthorised change is made? The discovery tool will certainly pick this change up and suddenly you have a corruption to the configuration. This is where my term 'enduring myth' comes in.

The 'myth' in this case relates to the widely held belief that once an i-CMDB is designed, populated, deployed and then automatically updated, this becomes the single authority at the core of a Configuration Management Sysstem. Also, this belief has been around for some years now – hence the term 'enduring'. I believe this approach introduces a major problem - one of over dependence on a technical solution. In viewing a federated CMDB as basically an automated solution at the core of a Configuration Management System there is a risk of supplanting the local knowledge and investigative experience of staff. This has obvious risks to service delivery.

Imagine, for a minute, that you are chairing a Change Approval Board (CAB) meeting where the i-CMDB impact report states that the proposed change has no impact on a business critical system and hence no further impact analysis is needed as the i-CMDB report is definitive and authoritative. Well, good luck with that and I might suggest you keep your CV up-to-date.

This appears cynical, but in the real world are we really going to hand over the key decisions for releasing a change based solely on an i-CMDB auto-generated report? If not, then why invest vast amounts of annual budget on attempting to do so? There is a paradox here and the way to solve it is to step back from attempting to implement an ITIL v3 compliant i-CMDB and look at the overall Configuration Management process and how we can improve on what we have by implementing a CMDB as part of a revised end-to-end Configuration Management process, but not necessarily the core.

A Question of Scale

Of course, in IT you can achieve almost any solution as long as you have deep pockets. In the real world most IT organisations are under pressure on budgets. The exception here maybe large organisations that have extensive IT dependency; for example a global investment bank with a complex infrastructure. Indeed, you can find some very good case studies on the IEEE Xplore Digital Library[2] about large scale CMDB installations that encompass 10,000 to 20,000 servers and a million plus CIs. Clearly, with this degree of scale there must be significant investment in an integrated CMDB along the design of ITIL SACM. However, in my view, these are exceptions and not the rule.

So, what are the options for a small to medium size IT organisation that wants to enhance their Configuration Management System to drive better service delivery, but have budget restraints? Well, here is my list.

Option 1 - live without a CMDB

Option 2 - leverage existing service desk CMDB functionality

Option 3 - design and build your own CMDB

Option 4 - buy a third party off the shelf product and do the implementation yourself

Option 5 - commission a vendor to design and implement the CMDB as a project

Option 1 - live without a CMDB

It is possible to operate a configuration management process without a CMDB, particularly if you are running a very small IT operation - and many organisations do. However, it really makes sense to capture and maintain all relevant CIs in one location, particularly if you are planning to build a Configuration Management System that is based on industry best practice.

Option 2 - leverage existing service desk CMDB functionality

This is more promising. Until quite recently, stand-alone Service Desk applications rarely included a CMDB as part of the functionality. However, most Service Desk software these days come as part of an ITSM tool set, and there is usually a CMDB included. This is sometime labeled a Configuration Management module, which may or may not be the same thing. There may also be an Asset Register that purports to be a CMDB as well, and so on. As there are dozens of ITSM tool sets available, it is difficult to be prescriptive here but you might be fortunate to have a Service Desk application that is part of an ITSM tool set that also has CMDB built-in. Often as not, this part of the ITSM functionality is left disabled or only part implemented.

More on this later.

Option 3 - design and build your own CMDB

I have already assumed that a multi-dimensional i-CMDB with all the associated interfacing will be difficult to implement so I would suggest that this is not even attempted by in-house design personnel. That said, if the CMDB is going to be a relational database design as previously described, then this is feasible as an in-house project. However, the downside means diverting key personnel for a design and build project that could take several months to implement. But it is a possible option.

Option 4 - buy a third party off the shelf product and do the implementation yourself

This is a minefield. There are numerous vendors that claim to offer CMDB solutions. Some explicitly state that their CMDB is a dimensional database and can be connected to your IT systems to provide all the federation, reconciliation, synchronisation and modelling needed to reach ITIL compliance. This may be so, but that is not our preferred approach as discussed previously.

There are also vendors who offer a CMDB relational database off-the-shelf and ready for inputting your CIs.

This could work but a word of caution. Out-of-the-box products often require third party consultancy to get them working correctly and the billable hours can clock up at an alarming rate. However, that said, it is a valid option and worth considering.

Option 5 **-Commission a vendor to design and implement the CMDB as a project**

Some vendors offer a bespoke CMDB but of course these will require a lot of expensive consultancy to implement fully, and few, if any, are that successful. The budget will be blown well before full implementation is achieved. This is my least favourite option as it has a high risk of failure.

So, which option(s) to choose? My first preference would be for Option 2. Look at what you already have included in your Service Desk software in the way of a relational database that will hold CIs. If it doesn't, then check with your Service Desk supplier in case there is a CMDB add-on module that is available.

Failing that, then I would look closely at Option 3 - build something yourself, and then Option 4 buy an off the shelf product.

Note: Should you be considering an upgrade to your Service Desk then this would present an ideal opportunity to factor in a CMDB as one of the core functions of the new Service Desk.

Selecting a CMDB

So, in summary, we are looking at a monolithic, rather than a federated CMDB. It is also likely to be a relational database model rather than a multi-dimensional CMDB. At this point I don't want to delve into the merits of relational databases versus multi-dimensional as this is outside of the scope of this paper. The key point I wish to make is that a relational database with multiple data tables built as sets of CI Records will provide a good working solution that combines semi-automatic and manual methods of controlling data changes to the CMDB.

This approach will depend on the database model of the Service Desk provider in Option 3 and the CMDB vendor in Option 4. If you choose Option 2, then you have the opportunity to research alternatives to the relational model approach. For the purpose of this paper I am assuming a relational database model.

Asset Register

Before I look at the steps for setting up of a CMDB, I want to touch on Asset Registers. These databases are often bundled with Service Desk software and sometimes vendors call them a CMDB. This is not true and an Asset Register is simply an inventory of hardware and software assets so that the ownership, purchase order history and licence management of assets can be tracked over the asset life-cycle - procurement to disposal. True, there will be some correlation with the CIs held in the CMDB and they will need to be kept synchronised.

Selecting the Configuration Item (CI)

So, what CIs do we need to capture to populate the CMDB? There are three main considerations and these are:

Consideration 1 -the scope of the CIs under control;

Consideration 2 - the attributes that need to be captured to create a CI Record

Consideration 3 - the relationship and dependencies between the various CIs

Consideration 1 -the scope of the CIs under control

What CIs do you actually want to control? If you can't control a CI then you can't change it and so it has no place in the CMDB. I believe it is best to start top down and start small.

As your Configuration Management System starts to stabilise and mature then add more CIs. Also, you should only be identifying CIs that are critical to the IT services you are providing. I'd certainly leave out including personnel (ideally located in HR System), documentation (ideally located in a KM System) and peripheral assets (ideally located in the Asset Register).

One possible way to restrict the number of CIs in your CMDB is to limit the CIs to those associated with Business Critical systems and some Non-Critical Systems. I would agree that this is not always possible as a lot of services are shared, but it is worth exploring. A possible sub-division might be made according to Severity:

1. All Business Critical and some non-Business Critical systems usually covered by Severity 1 and Severity 2 – locate in the CMDB;
2. All the remainder non Business Critical and also Legacy systems usually covered by Severity 3 and 4 – locate in a KM Database.

Consideration 2 – The attributes that need to be captured to create a CI record

The second consideration is the number of attributes that you want associated with each CI Record. Some common attributes are in the table below. Clearly, this is not exhaustive and a good source reference for identifying CI Attributes can be found in the ITIL Templates[3]

Note that CIs can be linked by relationship to ITIL processes, for example Incident Reports where the CI in question is associated to a particular service failure. Also, Service Reports where metrics can be collected about CI performance, like reliability. More on that later under Hot Spot scenarios.

No.	Description of Atribute
1	Unique Identifier
2	Classification
3	Description
4	Relationship to Incident Reports
5	Relationship to other CIs
6	Location
7	Dependencies
8	Manufacturer
9	Service Report
10	Modification History
n	Relationships to the IT Services
n+1	License Number

Details of the relationship between the CI and Incident Reportts (See Hot Spot 2)

History of MTTR and MTBR and other metrics for an individual CI (See Hot Spot 3)

Remember, the fewer attributes you have the easier it is going to be to maintain the CMDB. However, the attributes must be relevant enough to ensure that the CIs are identified correctly during the change management process. This trade off can only be resolved by the Configuration Management build team during the CMDB design phase.

Consideration 3 – The relationship and dependecies between the various CI Record

The third consideration is the relationship between the CIs. You need to know how the CIs are related to each other. Some typical questions about CIs are:

- Is it stand-alone?
- Is it an over-arching CI that has sub-ordinate CIs?
- Is it a component of something else – another CI?
- Is it a new version of an existing CI?
- Is it a replacement for an existing CI?
- Is it now redundant due to a change to the IT environment?t

This is a critical piece of work and identifying the dependencies between the CIs and building up a picture of the network of relationships that underpin the IT systems.

Populating the CMDB

Once the CIs have been collected as a set of CI Records we need to start to populate the CMDB. This is not a trivial exercise and must be a managed as a joint project between the Change Management and Configuration Management teams. Basically, the steps needed for the CI Record population are:

1. Freeze all current Change Requests (whilst a difficult task this will ensure that what you populate is an actual baseline configuration);
2. Create sets of CI Records that are divided into layers that relate to domain responsibility (this reduces the chance of erroneous updates to non-impacted CIs).
3. Perform one time population of some CI attributes using agentless auto-discovery tools (this will help speed the process);
4. Perform manual data uploads of the other CI attributes using pre-defined templates (these will have been compiled during the CI selection activity);
5. Perform a verification and audit on CMDB to ensure that everything has been captured.

Maintaining the CMDB

Now we have the baseline configuration it needs to be maintained and so monitoring and control of the CIs is critical. This must be the responsibility of the Configuration Managers, supported by Subject Matter Experts (SME's) for the IT domains of Server Infrastructure, Network Infrastructure and Application Management. Each domain is supported by various system monitoring and configuration tools, together with device management tools to extract the changes to the configuration data.

The CI change data is compiled as part of the normal Change Management process. The data is collected part manually (by SMEs) and part automatically (by configuration tools) to create a consolidated CI Data Table of changes that can be uploaded to the relevant CI Records within the CMDB, once the changes have been approved. This upload is initiated manually once the change is approved by the Configuration Management team. This method might appear a shade cumbersome but it has the advantage of ensuring a final 'human' decision to change the CMDB, as opposed to a fully automated one. The process is shown in Figure 3 on the next page of this document.

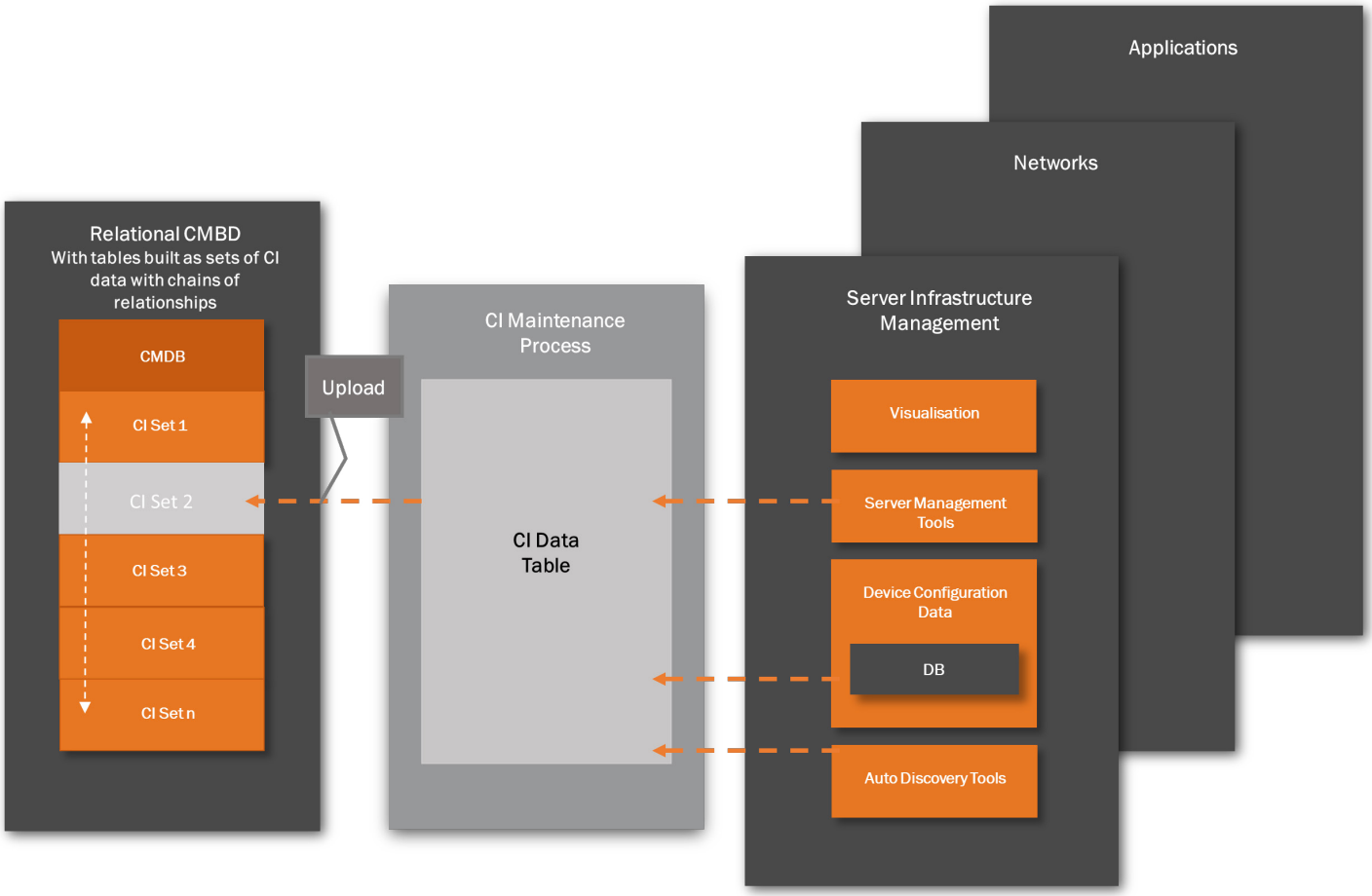


Figure 3 - Process for Maintaining the CMDB

Maintaining the CMDB

Now we have the baseline configuration it needs to be maintained and so monitoring and control of the CIs is critical. This must be the responsibility of the Configuration Managers, supported by Subject Matter Experts (SME's) for the IT domains of Server Infrastructure, Network Infrastructure and Application Management. Each domain is supported by various system monitoring and configuration tools, together with device management tools to extract the changes to the configuration data.

The CI change data is compiled as part of the normal Change Management process. The data is collected part manually (by SMEs) and part automatically (by configuration tools) to create a consolidated CI Data Table of changes that can be uploaded to the relevant CI Records within the CMDB, once the changes have been approved. This upload is initiated manually once the change is approved by the Configuration Management team. This method might appear a shade cumbersome but it has the advantage of ensuring a final 'human' decision to change the CMDB, as opposed to a fully automated one. The process is shown in Figure 3 on the next page of this document.

As mentioned, there will be a number of sources used for making a CI Record change:

System Management Tools - this is a very broad category and vendor products will vary considerably in functionality. Most, however, will provide data on device location, identification and status. Plus, of course Software Application Management (SAM) tools. These data will be monitored by the change management team.

Device Configuration Databases - this is usually a set of predefined databases containing configuration data for all configurable devices in the system. A record is kept of each device currently connected to a system and will be updated by an internal device configuration manager as changes are made.

Auto-discovery Tools – as mentioned previously, auto-discovery tools are best used for the initial population of the CI Records. For CMDB maintenance, auto-discovery tools cannot replace the manual and structured monitoring and control activities of experience staff. For example, auto-discovery tools are not good at detecting relationship changes between CIs. This must be done manually.

Visualisation - most management tools have some form of feature to visualise the relationship of components and dependencies. Although not a CI data source as such, this ability to map the devices will assist the configuration manager in making decisions on impact analysis, particularly when communicating with other configuration managers.

Designing the Configuration Management System

In the previous sections we looked at the two main types of CMDB - relational and multi-dimensional. I've suggested that a relational CMDB will work, but with some constraints. I've also looked at the various options for sourcing a CMDB and once sourced, provide suggestions on populating the CMDB with a selection of critical CIs, together with the appropriate attributes.

This will create a baseline configuration against which all future system and service changes can be made and recorded.

We now move on to designing our re-engineered Configuration Management System. Our next step is to put in place a Configuration Management System management team. A suggested list of key positions are:

- Lead Configuration Manager
- Server infrastructure Configuration Manager
- Network Infrastructure Configuration Manager
- Applications Management Configuration Manager

We are now ready to construct a Configuration Management System that uses the CMDB as a single source of reference for CIs. I'm only going to touch on the high level process and our starting point must be a Configuration Management Workflow that connects all the functions involved with a configuration change.

It's possible that a Configuration Management Workflow feature already exists within your Service Desk functionality. This may come bundled with the Incident, Problem, Request Fulfilment processes. If not, then it should either be possible to adapt one of the Service Desk other workflows to do the job, or alternatively there are a number of third party off-the-shelf workflow packages available.

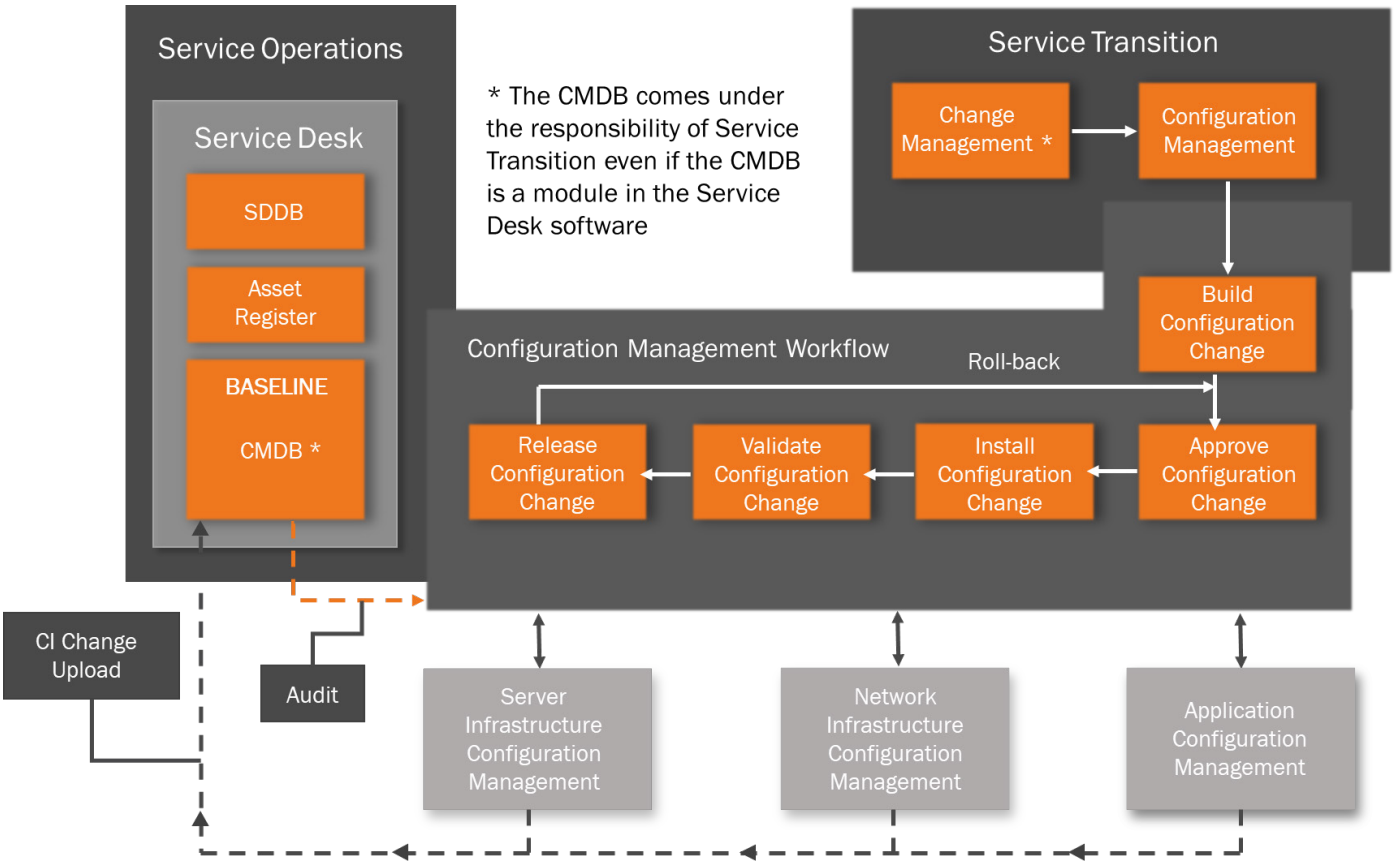


Figure 4 - Configuration Management System using a CMDB

However, once a change has been authorised then the Configuration Management Workflow will take the required change and trigger the actual technical build of the change to the CI (or groups of CIs). There then follows an Approval by the owners of the infrastructure which is impacted. Once this is granted the technical teams will Install and then Validate the change, ready for Release. A Roll-Back step is included should the change create a problem and the change needs to be reversed.

This is a normal Configuration Management System practice. However, as we now have a CMDB in place then once the change has been released the CMDB will need instant updating. In my view this is a red flag step. As explained under the Maintaining the CMDB heading, each of the Server, Network and Application teams must have a dedicated Configuration Manager who will be responsible for collating, reconciling and the uploading the change to the CI Layers in the CMDB. This must be done formally as part of the change release, not as an administration task to be sorted at a later date.

Maintenance of the integrity of the CMDB is the cornerstone of the Configuration Management System. Once the CI data becomes out of sync with the actual systems then the CMDB will become useless and give false impact reports. To drive this function there must also be an overall Configuration Manager (with authority) to oversee the whole end-to-end process and facilitate communication between the Change Manager and the technical teams. This is probably one of the most important roles to drive good Configuration Management practice.

A regular audit of the CMDB must be conducted by the Configuration Management team to ensure that the integrity of the configuration is maintained and reconcile any differences that might have occurred between the CIs as a result of the changes introduced since the last audit.

Example 'Hot Spots'

I've identified four typical 'hot spots' based on personal observations of real life events involving inadequate or erroneous system configuration data that have been the cause of major service outage.

These are shown in Figure 5. Clearly, there will be others depending on the set-up of a particular ITSM organisation and the types of client it supports.

Figure 5 is based on a simplified ITSM organisation that could be either a MSP dedicated to external clients, or an ITSM organisation providing IT services to an internal client. The IT Operations can be either internal or external hosting with or without applications support. It assumes that all key components are under the control of the IT organisation.

For the purpose of this paper it is assumed that the IT Operations is in-house and provides hosting, communications and applications support - within an overall governance framework.

There are four example 'hot 'spots' shown in Figure 5.

- **Hot Spot 1 (Change Management)** – Risk to service delivery due to poor change control
- **Hot Spot 2 (Incident Management)** – Risk of service failure due to poor incident resolution time
- **Hot Spot 3 (Availability Management)** – Risk to service recovery due to extended CI repair times
- **Hot Spot 4 (Service Continuity)** – Risk to service continuity due to lack of data on current system configuration

All of the above examples involve insufficient knowledge of current IT system configuration and the following narrative will describe how the implementation of a more robust Configuration Management System with a CMDB can help mitigate these risks.

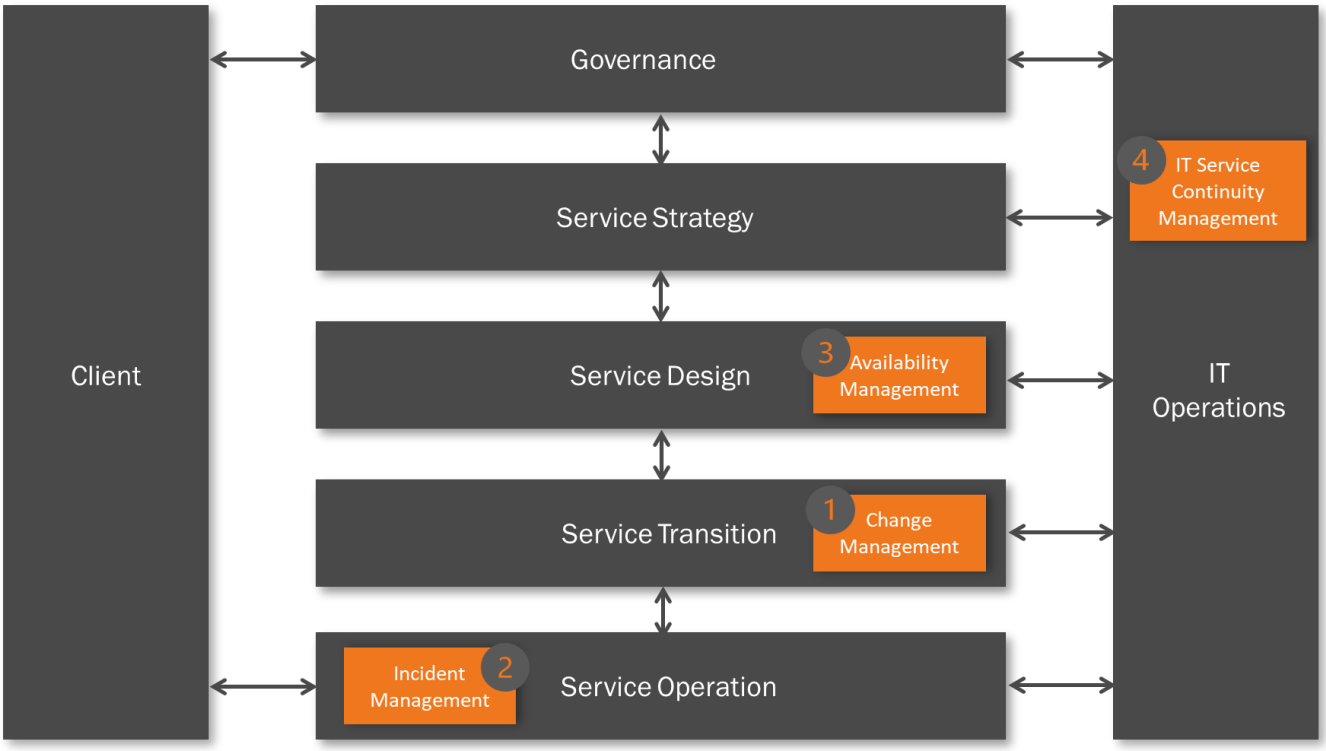


Figure 5 - Example Hot Spots

Hot Spot 1 - Risk to Service Delivery

The 'Risk' we are trying to mitigate here is one of a poorly managed change request that leads to an uncontrolled service outage and associated breach in SLAs. Typically, this is due to inadequate Impact Analysis resulting from:

- an incomplete baseline for the current system configuration;
- insufficient understanding of dependencies and relationships between the CIs;
- unclear ownership of CIs;
- poor communication between technical teams.

The implementation of a CMDB as outlined in this paper will eliminate many of the above deficiencies, but of course no technology solution on its own will compensate for poor teamwork and team communications. In this scenario I've created a basic Change Management process to demonstrate how these risks to service delivery can be mitigated by using a CMDB as the source of the current baseline, plus supporting analysis from technical teams based on ITSM tool sets and local experience from SMEs.

Figure 6 shows a simplified Change Management process with a Change Advisory Board as the forum for bringing together the various inputs for analysis.

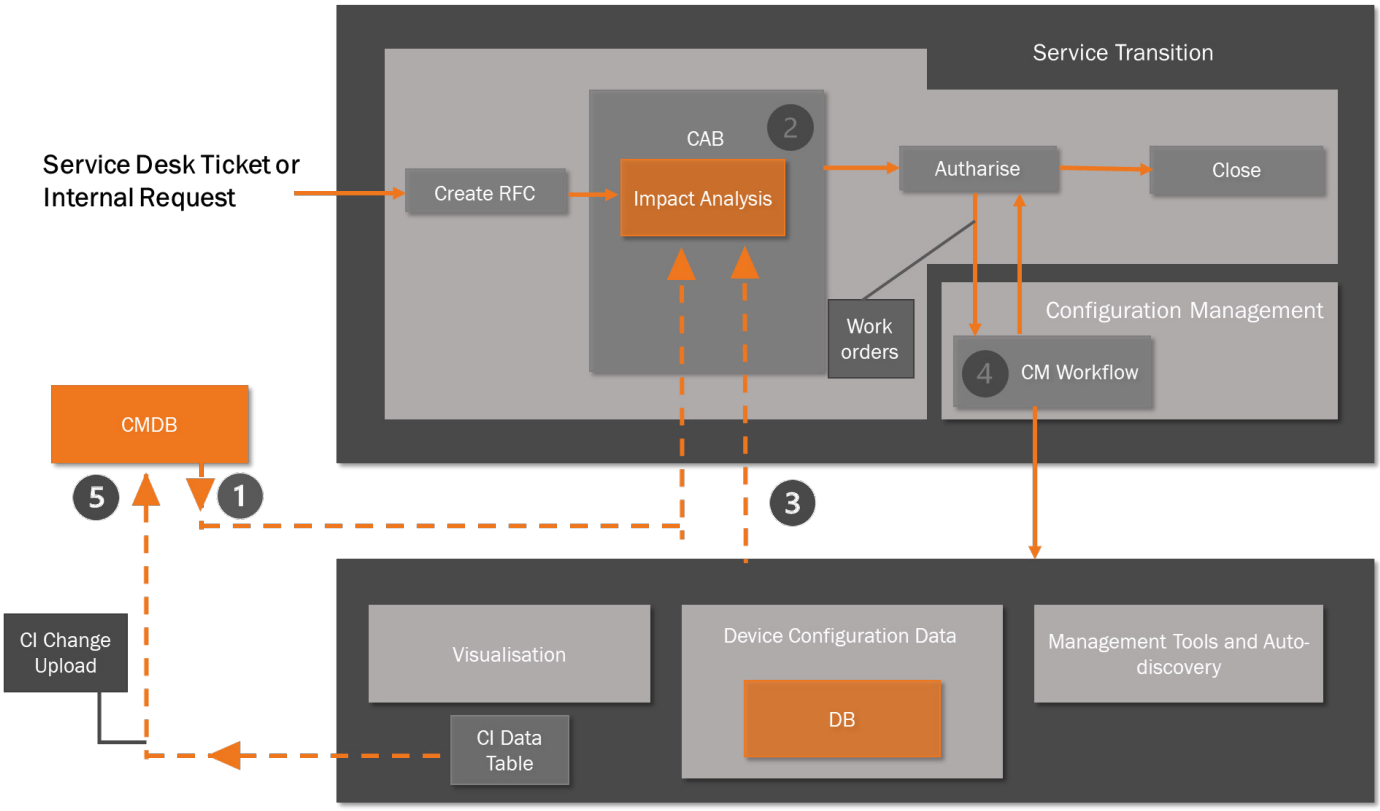


Figure 6 - Reducing Risk to Service Delivery

For the purpose of this scenario I'm concentrating on how the CAB will operate with the SMEsusing the various sources of data available to them when analysing the impact of a Request for Change (RFC). There are five key steps in this process:

1. The Change Manager will run a report from the CMDB that shows a baseline for the current configuration for that part of the IT environment under proposed change.
2. This is expressed as a set of Configuration Items (CI) with known attributes that can be used as the starting point for the Impact Analysis process.
3. The CAB comprises a group of SMEs tasked with evaluating the proposed change to the IT system - i.e. Impact Analysis. These will typically be drawn from the server, network and application infrastructure and design teams, service owners, IT Security and IT Operations.
4. The SMEs will use a variety of data sources together with the CMDB baseline report to determine how the CIs will be impacted in terms of the relationships and dependencies with other CIs. The data sources, already discussed, will be outputs from:

- Visualisation tools, for example Server node modelling;
- Network device management tools;
- Software Asset Management (SAM) tools that can identify dependencies between software modules;
- Auto-discovery tools;
- Plus of course a considerable amount of SME experience to interpret and model all possible change
- scenarios likely to impact the IT environment.

5. Once the analysis is complete the Change Manager will authorise the change to be madevia Work Orders to the appropriate technical teams. There will also be an agreed roll-back plan should the change fail. This is managed by the Configuration Management team.
6. The CM Workflow will be used to manage the Work Orders through to completion. The final step is for all the changes to the CIs to be collected and uploaded to the CMDB to create the current configuration baseline resulting from the change.

Hot Spot 2 - Risk to Service Failure

For the second Hot Spot I want to look at how our Configuration Management System can be used to reduce the time taken to resolve incidents. I also want to look at how the incident records can be linked to the associated CIs held in the CMDB to aid this process.

Incidents can be any type of failure or interruption to an IT service. Incidents are created from a number of sources, like customers' phone calls and technical staff alerts. In this scenario all the Incidents are routed through the Service Desk where the incident is logged and an incident ticket raised.

The incident ticket will contain a brief description of the incident and its impact on services or systems. This is usually added manually by the IM team. The majority of incidents can usually be handled and cleared by the IM team without consulting the technical teams.

However, in our scenario the IM Process is integrated with a Configuration Management System that has a CMDB as the source of CI Records. This will provide the IM team an extra source of data when SMEs are required to diagnose the root cause of an incident. Figure 7 details a simple IM process and how this links with the Configuration Management System.

1. The Incident Manager will run a report from the CMDB based on what is known about the incident. For example, the service impacted, the location, known hardware or software failure. The report will list all the CIs and dependant CIs that have these attributes in their CI Records. The report is then attached to the incident ticket. It is also possible to add in a further attribute to a CI Record that links known incident types to certain CI(s) and dependent CIs. See Step 4 below.
2. The incident ticket is then Categorised – by priority and urgency – and passed to the Diagnose stage where it is routed to the correct SME team for analysis. (Note: Attaching an initial CMDB report will help to route the incident to the correct SME(s) first time round rather than the incident being passed from one SME to another due to wrong allocation.
3. The SMEs will conduct diagnostics using the various tools sets already outlined, namely Visualisation tools, Device Configuration tools and System Management and Auto-discovery tools. Once the diagnosis is complete the SMEs will notify the IM team that the incident is now resolved.
4. It is possible to associate CIs to an incident. This can be done by adding an attribute – Relationships to Incident Records to each CI Record. This will help build up a picture of how types of incidents effect certain CIs and other CIs with dependent relationships. The SME team can update this CMDB attribute direct without going through the Configuration Management team.

Over time, collecting this feedback on the relationship between incident types and particular CIs will help speed future incident resolution and provide system designers with potential system upgrades to improve performance.

5. The resolution may involve an update or replacement of hardware, or infrastructure, which in turn will result in a Change Request. The CM Team will action this change and update the CMDB as outlined previously under the heading - *Designing a Configuration Management System.*

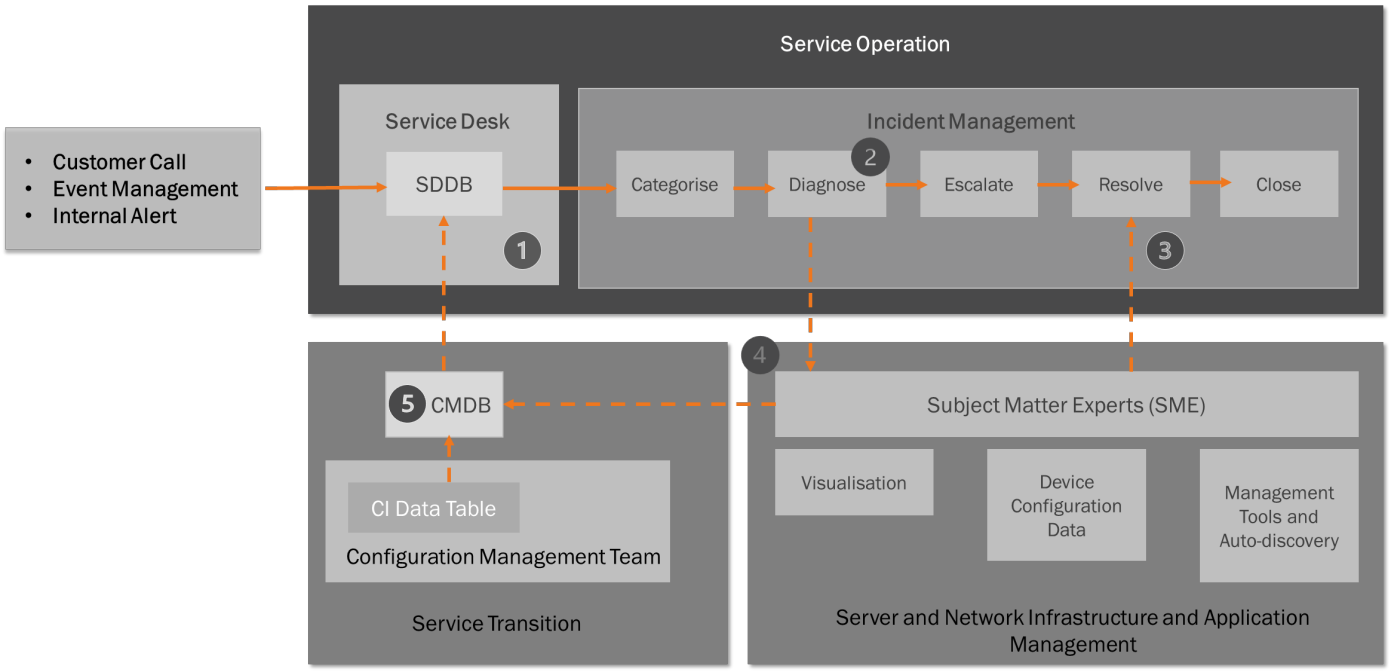


Figure 7 - Reducing Risk to Service Failure

Hot Spot 3 - Risk to Service Recovery

For my third 'Hot Spot' I want to look briefly at how a CMDB can assist with Availability Management and the risk to service recovery due to extended CI repair times.

Availability is the amount of time a service is actually operating or a percentage of total time it should be operating. There are two other terms we need to consider here - Reliability and Maintainability.

- The reliability of a CI indicates how long it can perform its agreed function without interruption.
- The maintainability of a CI indicates how fast it can be restored after failure.

The time between system incidents is a factor of maintainability and reliability. Considering most business critical systems are designed as high-availability and as such will have hardware redundancy and failover, then the more data that can be collected relating to the individual CIs the greater the understanding we have of both system design and CI selection for that design.

This is key to service improvement and it is possible for the CMDB to hold data on the individual CI performance life cycle. One way we can do this is to collect metrics around CIs that make up our system or service. This will aid our understanding of both the reliability and maintainability of those CIs. There are two metrics that we can collect:

- Mean Time between Failure (MTBF) - is the average time that a CI can perform its agreed function without interruption. This is uptime - Reliability.
- Mean Time to Repair (MTTR) - is the average time taken to repair a CI after a failure. This is downtime - Maintainability.

So, one way we can collect this data is to add an attribute to our CI Record. This is a Service Record field and will hold data on both MTBF and MTTR for individual CIs.

This performance data is collected and added to the CMDB as part of the Configuration Management process that following the closure of an incident. See Hot Spot 2 above.

Hot Spot 4 - Risk to Service Continuity

For my third 'Hot Spot' I want to look briefly at how a CMDB can assist with Availability Management and the risk to service Throughout this paper I have built up a picture of how a CMDB can be used as a key component in a Configuration Management System. Over time, the data held in CI Records will build into a repository that will define the current system and service configuration as expressed in terms of a range of CI attributes including all dependencies. In this final scenario I want to look at how the CMDB can be used to support Service Continuity.

When discussing Service Continuity there are multiple terms involved – like Disaster Recovery, Business Recovery and Service Recovery and I'm not intending to cover the whole subject of Business Continuity Management (BCM). However, IT organisations will to some degree need to build IT Disaster Recovery Plans or Business Continuity Plans to meet general governance or specific client requirements. These plans should be structured along the scope and guidelines of ISO22301[4] the closure of an incident.

To support this planning activity we now have three principle sources of data:

1. CMDB – that defines the current system and service configuration in terms of CIs and dependencies;

2. Asset Register – that shows all the data centre assets not held in the CMDB

3. KMDB – that holds all the business continuity plans and recovery procedures

Collectively, all this information will form the core repository of assets and configuration that are needed to plan for disruptive incidents and thus meet the Recovery Time Objective (RTO). The RTO is the period of time following an incident within which a service must be resumed or resources recovered.

Clearly, there is much more to discuss here, but there is a future White Paper that will address all of these aspects of service recovery. This is planned for publication later in 2015.

Conclusion

This paper looks at how to re-organise an existing Configuration Management System (CMS) by including a CMDB as a central location for all CIs principally associated with business critical services and systems.

The emphasis has been on:

- using what already exists within a Service Desk software suite or ITSM tool set and utilising those features;
- building or purchasing a CMDB solution as described in the various options;
- adapting existing processes and policies to accommodate the CI data in the CMDB;
- introducing a Configuration Management Workflow solution;
- support by an experienced team of configuration managers and SMEs.

These steps, although not all encompassing, will enable the Change Management and Configuration Management teams to minimise the risks involved when changing the configuration of an IT system.

References:

1 Gartner Report - Critical Capabilities for Configuration Management Database, published in 2014

2 IEEE Xplore Digital Library – Best Practices for Developing a CMDB in large Scale Environments 2009

3 Checklist Configuration Item (CI) Record – http://wiki.en.it-processmaps.com

4 ISO 22301 – Business Continuity Management Systems.